NASA-CR-192745

CIRSSE DOCUMENT #45

NAGW-1333

4 99

TECHNICAL REPORTS
UNIVERSITY COLLECTION

TECHNICAL REPORTS

[illegible]
IN-33-CR
153777
p. 60

(NASA-CR-192745) THE APPLICATION OF THE REDUCED ORDER MODEL KALMAN FILTER TO MOTION ESTIMATION OF DEGRADED IMAGE SEQUENCES M.S. Thesis (Rensselaer Polytechnic Inst.) 60 p

N93-71642

Unclas

Z9/33 0153777

CIRSSE

Center for Intelligent Robotic Systems for Space Exploration

Rensselaer Polytechnic Institute
Troy, New York 12180-3590

Technical Reports
Engineering and Physical Sciences Library
University of Maryland
College Park, Maryland 20742

THE APPLICATION OF THE REDUCED ORDER MODEL KALMAN FILTER TO MOTION ESTIMATION OF DEGRADED IMAGE SEQUENCES

By:

Elizabeth C. Simpson

Department of Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

CIRSSE Document #45

THE APPLICATION OF THE REDUCED ORDER MODEL KALMAN FILTER TO MOTION ESTIMATION OF DEGRADED IMAGE SEQUENCES

by

Elizabeth C. Simpson

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Approved:

[illegible]

Dr. Howard Kaufman
Thesis Advisor

Rensselaer Polytechnic Institute
Troy, New York

December 1989

CONTENTS

LIST OF TABLES iii

LIST OF FIGURES iv

ACKNOWLEDGEMENT v

ABSTRACT vi

1. Introduction 1

2. The ROMKF 4

    2.1 Introduction 4

    2.2 Theory 5

    2.3 Experimental Results 9

    2.4 SUN Implementation 16

3. Motion Estimation 19

    3.1 Introduction 19

    3.2 Theory 19

    3.2 Experimental Results 22

4. Discussion and Conclusions 42

A. Blur and State Supports 45

B. The Kalman Filter Equations 50

REFERENCES 52

LIST OF TABLES

Table 2.1 Results for 1 x 5 Blur with Model Size Varied....11

Table 2.2 Results for 1 x 5 Blur with State Size Varied....12

Table 2.3 Results for 3 x 3 Blur with Model Size Varied....13

Table 2.4 Results for 3 x 3 Blur with State Size Varied....14

Table 3.1 Results of Varying the Number of Pyramid Levels....25

Table 3.2 Filter Parameters....27

Table 3.3 Results with 20 dB Noise....28

Table 3.4 Results with 1 x 5 Blur....29

Table 3.5 Results with 1 x 5 Blur plus 20 dB Noise....30

Table 3.6 Results of Thresholding the Circle Displacements (Rest. Sequence), 1 x 5 Blur Case....37

Table 3.7 Filtered Displacement Improvements....38

LIST OF FIGURES

Figure 2.1 $M_1 \times M_2 \times M_3$ NSHP Support 6

Figure 2.2 Unreduced State Support 7

Figure 2.3 ROM State Support 8

Figure 2.4 Original Cameraman Image 9

Figure 2.5 Blur Supports Used 10

Figure 2.6 Comparison of Minimum State with Cases with Maximum Improvement - 1 x 5 Blur, 40 dB BSNR 15

Figure 2.7 Comparison of Minimum State with Cases with Maximum Improvement - 3 x 3 Blur, 40 dB BSNR 17

Figure 3.1 Subbands Used in the QMF Pyramid 21

Figure 3.2 Pyramid Construction 21

Figure 3.3 Artificial Circle Image 23

Figure 3.4 Robot Arm Image 24

Figure 3.5 NHSP Support 27

Figure 3.6 Degraded and Restored Circle Images 32

Figure 3.7 Circle Displacements for 20 dB Noise 33

Figure 3.8 Circle Displacements for 1 x 5 Blur 34

Figure 3.9 Circle Displacements for 1 x 5 Blur plus 20 dB Noise 35

Figure 3.10 Displacements for 10 Pixel xy velocity, 1 x 5 Blur 39

Figure 3.11 Degraded and Restored Robot Arm Images 40

Figure 3.12 Robot Arm Displacements for 20 dB Noise 41

ACKNOWLEDGEMENT

My work on this thesis was helped enormously by many individuals First, I would like to thank Prof. Howard Kaufman for selecting me as a research assistant, and for all his help which he has given during my time at RPI. I would also like to thank Denise Angwin for being so patient in helping me understand her work, which is the basis of this thesis .

Many people in the IPL gave me valuable advice; in particular, Bob Cohen, T. Naveen, Y. Kim, and Doug Lyon. I would especially like to thank A.J. Lacomba for going out of his way to try and keep the IPL running smoothly.

I would like to thank all my friends here at RPI who have been there when I needed advice, entertainment, and most importantly, rides home. Special thanks go to Steve Blaes who helped to proofread this thesis.

This work was supported by the following: NASA grant NAGW-1333, a grant from the Eastman Kodak Research Laboratories, and the NSF institutional infrastructure grant CDA-8805910.

Finally, I would like to thank my parents for their constant support and encouragement of the decisions that I have made in my life, no matter how unexpected. I also want to thank Mike for putting up with my frustrations and always giving me something to look forward to.

ABSTRACT

Motion estimation is a field of great interest because of its many applications in areas such as robotics and image coding. The optic flow method is one such scheme which, although fairly accurate, is prone to error in the presence of noise. This thesis describes the use of the reduced order model Kalman filter (ROMKF) in reducing errors in displacement estimation due to degradation of the sequence. The implementation of filtering and motion estimation algorithms on the SUN workstation is also discussed.

Results from preliminary testing were used to determine the degrees of freedom available for the ROMKF in the SUN software. The tests indicated that increasing the state to the left leads to slight improvement over the minimum state case. Therefore, the software uses the minimum model, with the option of adding states to the left only.

The ROMKF was then used in conjunction with a hierarchical pel recursive motion estimation algorithm. Applying the ROMKF to the degraded displacements themselves generally yielded slight improvements in cases with noise degradation and noise plus blur. Filtering the images of the degraded sequence prior to motion estimation was less effective in these cases. Both methods performed badly in the case of blur alone, resulting in increased displacement errors. This is thought to be due in part to filter artifacts. Some improvements were obtained by varying the filter parameters when filtering the displacements directly. This result suggests that further study in varying filter parameters may lead to better results.

The results of this thesis indicate that the ROMKF can play a part in reducing motion estimation errors from degraded sequences. However, more work needs to be done before the use of the ROMKF can be a practical solution.

PART 1

Introduction

Estimating the motion of objects in a scene is a problem being widely researched today. Motion estimation has a wide range of applications in the fields of image processing and computer vision. For example, motion estimation can be used in image coding to decrease the bandwidth needed to transmit an image sequence. In the area of robotics, velocities of objects can be used for navigation and guidance purposes. In addition, motion estimation can be valuable in the area of medical imaging in observing the motion of the heart. Other uses include satellite weather tracking and traffic surveillance.

Motion estimation methods, as discussed in the survey by Aggarwal and Nandhakumar [1], can be grouped into two basic categories: feature-based estimation, and optic flow computation. In feature based estimation, a set of two-dimensional features is first extracted, corresponding to three-dimensional object features in the scene. Examples of such features are corners, occluding boundaries of surfaces, and boundaries between areas of differing surface reflectivity. Next, inter-frame correspondence is established between the features, with constraints based on assumptions about the motion, such as rigid body motion. Finding such a correspondence is difficult, and only partial solutions for simple situations have been developed. The resulting equations for the motion are then solved using the observed displacements of the two-dimensional features.

The optic flow method consists of computing the velocity of each pixel in the image, resulting in a two-dimensional field of instantaneous velocities. Unlike the feature-based approach, optic flow calculations do not require that any feature correspondence be established. Instead, changes in image brightness can be used to compute the optic flow. It is also possible to use changes in variables other than brightness which result from applying local operators such as contrast, entropy, or spatial derivatives. Many examples of optic flow algorithms can be found in [1].

Once estimated, optic flow can be used to calculate depth, as in the paper by Matthies, Szeliski, and Kanade [7] , in which a Kalman filtering approach is used to estimate depth from motion, recursively refining the estimate over time. Another extension of optic flow motion estimation is in the calculation of rigid body motion from depth and optic flow, as in the paper by Ballard and Kimball [3] . In this case, optic flow and depth information are used to compute the nine parameters which completely describe a rigid body's motion in terms of the position, translational velocity, and rotational velocity of the body's local coordinate origin.

The relative simplicity of the optic flow estimation as well as its extensions to three-dimensional motion make it an attractive method for many applications. However, as discussed by Aggarwal and Nandhukumar [1] , optic flow computation is prone to error in the presence of noise. The computation of the optic flow depends on taking partial derivatives of the image brightness values, and the evaluation of derivatives is a noise enhancing process. Therefore, it is of value to investigate methods of decreasing the error in motion estimation due to degradations of the images. It is also useful to have a software tool with which these and other image processing methods can be implemented in a user-friendly environment. This thesis explores the usage of the reduced order model Kalman filter (ROMKF) in reducing motion estimation errors, and discusses the implementation of the algorithms on the SUN workstation.

The ROMKF was developed by Angwin [2] to reduce the number of computations needed to implement the Kalman filter in image restoration. Since many motion estimation applications such as robotic navigation require real-time computations, the ROMKF is a good choice because of its speed. In addition, the form of the ROMKF is well suited to use in filtering the optic flow fields as well as the images of the sequences. The development of the ROMKF is discussed in Part 2 of the thesis, as well as results of preliminary studies on the optimum model/state sizes, which affected the final SUN implementation. Part 3 discusses the motion estimation algorithm, and compares results of filtering the sequences prior to the optic flow calculations to filtering the displacement fields themselves. Finally, Part 4 presents conclusions and recommendations for further study.

PART 2
The ROMKF

2.1 Introduction

Images are generally not perfect reproductions of the scenes they represent, but rather are degraded by the sensing process. For example, the image may be corrupted by sensor noise, or blurred due to relative motion of the camera and object or camera misfocus. Image restoration is concerned with recovering the original scene from the degraded image by removing the degradation. Image restoration is therefore different from image enhancement, which concentrates on extracting and accentuating certain features of an image without regard to removing the degradation itself.

One image restoration method which has been developed from estimation theory is the Kalman filter. The Kalman Filter is a recursive filter using a state variable representation of the system, which in this case is an image. One difficulty with the Kalman Filter implementation is the high order of the state vector that is required, necessitating large amounts of memory and extensive computations. Consequently, various algorithms have been developed to decrease the number of computations needed for implementing the Kalman Filter in image restoration applications. For example, Woods and Radewan [11] extended the Kalman filter to two dimensions, proposing the Reduced Update Kalman Filter (RUKF). Another algorithm by Mahalanabis and Xue [6] uses a two-dimensional Chandrasekhar filter, another vector Kalman filter. The ROMKF is a scalar filter in which a lower order state vector is used to reduce the number of computations required. The ROMKF has been shown to have comparable performance to both the RUKF and the Chandrasekhar filter [2], using the minimum state and model sizes. The purpose of this study was to vary the state and model sizes to see if an improvement is possible over the minimum case.

2.2 Theory

The ROMKF algorithm is based on an image model of the following form:

$$s(m,n) = \sum_{k,l \in R_1} c_{kl}(m,n)s(m-k,n-l) + w(m,n), \tag{2.1}$$

where (m,n) refers to the pixel in the mth row and nth column, $s(m,n)$ is the original image, $c_{kl}(m,n)$ represent the model coefficients, $w(m,n)$ is gaussian white noise accounting for the error in the model, and R_1 is the nonsymmetric half plane (NSHP) model support shown in figure 2.1. The observed, degraded image is modeled as the output of a linear filter as follows:

$$r(m,n) = \sum_{i,j \in R_2} h_{ij}(m,n)s(m-i,n-j) + v(m,n) , \tag{2.2}$$

where $r(m,n)$ is the observed degraded image, $s(m,n)$ is the original image from (2.1), $v(m,n)$ is measurement noise, and $h_{ij}(m,n)$ represent the spatially varying degradation point spread function (PSF) with support R_2.

The Kalman filter is a recursive spatial domain estimator which estimates the original image s, given the observations r for each image pixel. The filter equations are given in Appendix B. To use the Kalman filter, the models for s and r must be incorporated into a state-space representation of the form:

$$\underline{x}(m,n) = C\underline{x}(m-1,n) + E\underline{u}(m,n) + D\underline{w}(m,n) \tag{2.3}$$

$$r(\mathrm{m,n}) = H\underline{x}(m,n) + v(m,n) . \tag{2.4}$$

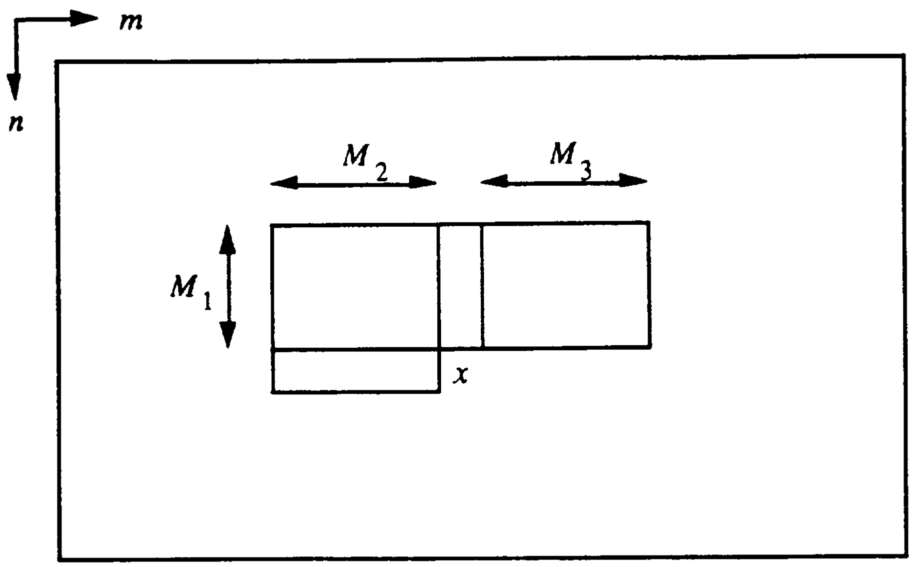


Figure 2.1: M_1 x M_2 x M_3 NSHP Support

Here, $\underline{x}(m,n)$ is the state vector at pixel (m,n), $\underline{u}(m,n)$ is a deterministic input, and $\boldsymbol{C}, \boldsymbol{D}$, and $\boldsymbol{H}$ are system matrices. The terms r, v, and $\underline{w}$ are as previously defined. This representation is a one-dimensional state-space difference equation, meaning there is one direction of state propagation. In raster scan format, which is used in this algorithm, horizontal propagation is used from left to right until a boundary is reached. At the boundary, the vertical index is incremented, and the horizontal index is reset to the left boundary.

In full-state Kalman image filtering, the state is defined as follows:

$$\begin{aligned}\underline{x}(m,n) = [\ & s(m,n), s(m-1,n),\ldots, s(1,n); \\ & s(N,n-1), s(N-1,n-1),\ldots, s(1,n-1); \\ & \ldots; \\ & s(N,n-M_1), s(N-1,n-M_1),\ldots, s(m-M_2,n-M_1)\ ]^T .\end{aligned} \tag{2.5}$$

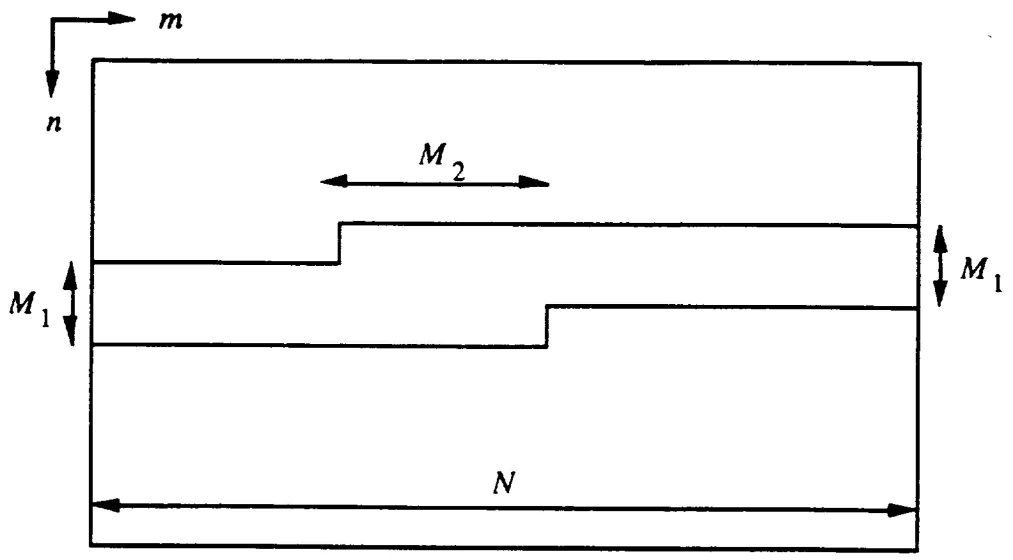


Figure 2.2: Unreduced State Support

This definition applies to an image of N pixels in width that is scanned from left to right, top to bottom. The state support is shown in figure 2.2. For this state representation, the $\boldsymbol{E}$ matrix is the null matrix.

Using the state as defined above would require excessive computational effort, as the state size is $O(M_1N)$. It was therefore proposed in [2] to reduce the size of the state vector by using only those pixels required to represent the image models of (2.1) and (2.2). To do this, it is necessary to make approximations for pixels just to the right of the M_1 x M_2 x M_3 model support. This reduced order state vector, shown in figure 2.3, is as follows:

$$\begin{aligned}\underline{x}(m,n) = [\ & s(m,n), s(m-1,n), \ldots, s(m-M_2,n);\\ & s(m-M_2,n-1), s(m-M_2+1,n-1), \ldots, s(m+M_3+1,n-M_1);\\ & \ldots \\ & s(m-M_2,n-M_1), s(m-M_2+1,n-M_1), \ldots, s(m+M_3+1,n-M_1)]^T.\end{aligned} \tag{2.6}$$

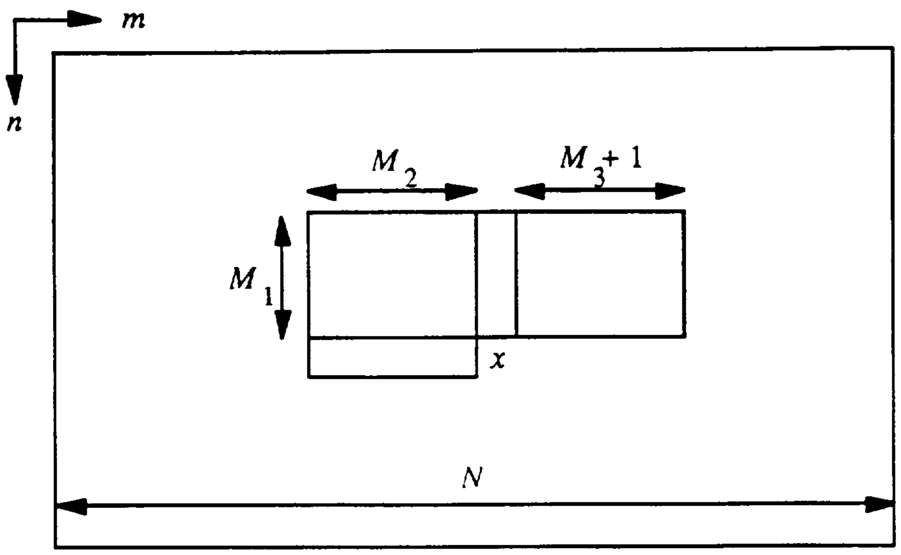


Figure 2.3: ROM State Support

A major feature of the ROMKF is the treatment of the pixels in the state $\underline{x}(m,n)$ which cannot be represented in terms of the shifted state $\underline{x}(m - 1,n)$. In the ROMKF, these pixels are approximated by their most recent estimates, with the uncertainty represented in a noise term. This is shown by the following equation:

$$s(m + M_3 + 1, \cdot\,) = \hat{s}(m + M_3 + 1, \cdot\,) + w_2(m,n)\ , \tag{2.7}$$

where $\hat{s}(\cdot, \cdot)$ is the most recent update of the pixel available at the time pixel (m,n) is filtered. This approximation is included in the state equation as the deterministic input $\underline{u}(m,n)$ from (2.3).

2.3 Experimental Results

The ROMKF was applied to the cameraman image of size 128 by 128 pixels which has been degraded by known blur and noise. The original image is shown in figure 2.4. Two different blur supports were used: a 1 x 5 linear blur , and a 3 x 3 two-dimensional blur. These blur supports are shown in figure 2.5. Two cases for the added noise were also used: 30 dB blurred SNR, and 40 dB blurred SNR, where blurred SNR (BSNR) is given as:

$$\mathrm{BSNR}_{dB} = 10\log_{10}\frac{\text{blurred image variance}}{\text{observation noise variance}} \quad . \tag{2.8}$$

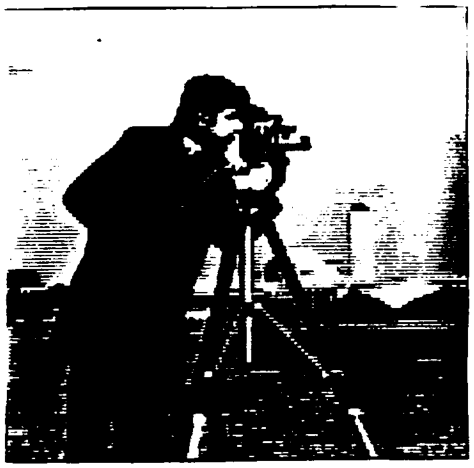


Figure 2.4: Original Cameraman Image

The filter was then implemented using varied model supports and varied state supports as shown in Appendix A. The model parameters were estimated using recursive least squares parameter estimation on the original unblurred image before filtering. All software was adapted from programs written by Angwin [2] and implemented on the Prime and Michigan Terminal System mainframes.

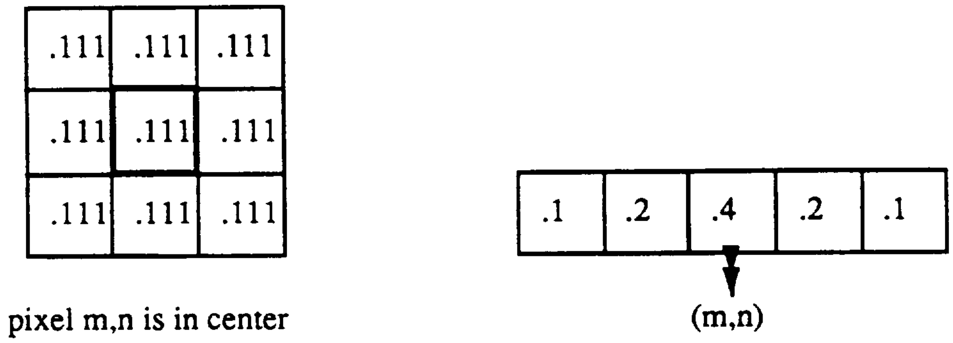


Figure 2.5: Blur Supports Used

The results are presented in the form of the mean-square-error improvement, η, calculated in decibels in the following manner:

$$\eta_{dB} = 10\log_{10} \frac{\Sigma(s(m,n) - r(m,n))^2}{\Sigma(s(m,n) - \hat{s}(m,n))^2} \quad . \tag{2.9}$$

The improvements are displayed in tables 2.1-2.4. These results show that increasing the model size in either the vertical or right direction (M_2 or M_3) has little effect on filter performance when comparing with the minimum model, minimum state case. Similarly, extending the state support in these directions also has little effect. However, increasing either the model or state support region in the left direction can improve the result. This is most likely due to the increased smoothing arising from a larger update area.

Table 2.1: Results for 1 x 5 Blur with Model Size Varied

Case	Model Size M_1xM_2xM_3	MSE Improvement η_{dB} 30 dB noise	40 dB noise
1	1 x 1 x 1	4.75	10.60
2	1 x 1 x 2	4.78	10.68
3	1 x 1 x 6	4.88	10.76
4	1 x 6 x 1	5.01	12.81
5	1 x 6 x 6	5.11	12.97
6	2 x 1 x 1	4.75	9.97
7	2 x 2 x 1	4.85	9.74
8	2 x 2 x 2	4.83	9.72
9	3 x 3 x 3	4.36	6.52

Table 2.2: Results for 1 x 5 Blur with State Size Varied

Case	Pixels added	MSE Improvement η_{dB} 30 dB noise	40 dB noise
1	1 - to right	4.76	10.57
2	5 - to right	4.76	10.57
3	10 - to right	4.75	10.57
4	1 - to left	4.92	12.04
5	5 - to left	4.90	12.77
6	10 - to left	4.89	12.78
7	5 - right, 5 - left	4.89	12.77
8	10 - right, 10 - left	4.89	12.78
9	1 - vertically	5.14	9.45
10	1x1x2 model, extra states	5.13	9.18
11	2x1x2 model, extra states	5.09	9.15
12	2x2x2 model, extra states	5.07	9.16
13	1x1x1 model, extra states	4.66	7.10

Table 2.3: Results of 3 x 3 blur with Model Size Varied

	Model Size	MSE Improvement η_{dB}	
Case	M_1xM_2xM_3	30 dB noise	40 dB noise
1	1 x 1 x 1	2.47	5.04
2	1 x 1 x 2	2.46	5.04
3	1 x 1 x 6	2.76	4.88
4	1 x 6 x 1	3.63	5.99
5	1 x 6 x 6	3.66	6.07
6	2 x 1 x 1	2.63	4.88
7	2 x 2 x 2	2.65	4.95
8	3 x 3 x 3	1.72	2.17

Table 2.4: Results for 3 x 3 Blur with State Size Varied

Case	Pixels Added	MSE Improvement η_{dB} 30 dB noise	MSE Improvement η_{dB} 40 dB noise
1	1 - to right (1 row)	2.47	5.04
2	5 - to right (1 row)	2.61	5.04
3	5 - to right (2 rows)	2.39	4.82
4	1 - to left (3 rows)	3.36	5.72
5	5 - to left (3 rows)	3.63	6.27
6	5 - to left (1 row)	3.40	5.90
7	10 - to left (3 rows)	3.71	6.38
8	5 - to right (2 rows), 5 - to left (3 rows)	3.68	6.19

For the cases used, the maximum improvement was approximately two dB. This was seen in the 1 x 5 blur, 40 dB noise case. The subjective improvement of these images as determined by visual inspection was at most very slight; in most cases, improvement was imperceptible. Examples of these restored images are compared with the degraded image in figure 2.6. The mean-square improvement for the 30 dB noise case was less, and the visual results were similar to the above cases. These slight improvements were obtained at greater computational cost due to the added model parameters and/or states, as compared with the lowest order 1 x 1 x 1 model with nine states.

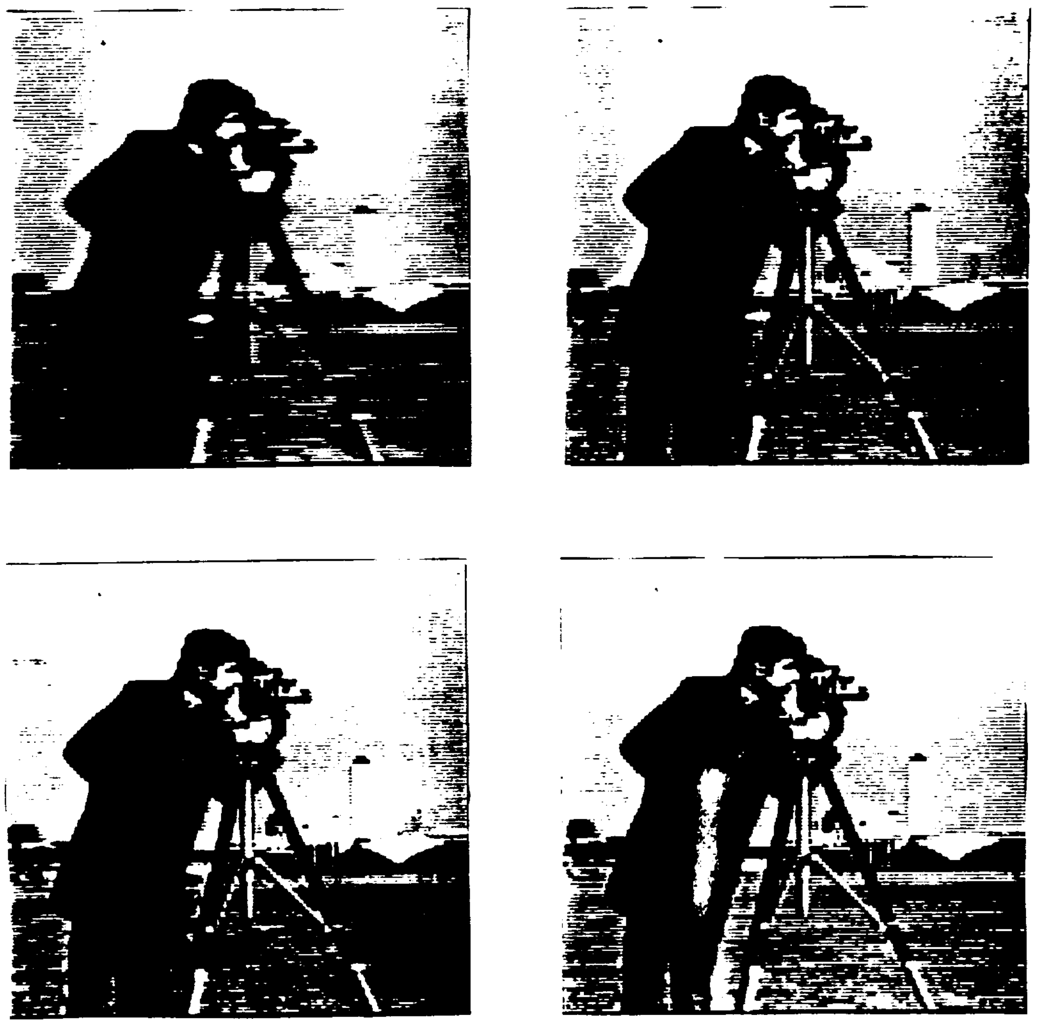


Figure 2.6: Comparison of Minimum State with Cases with Maximum Improvement - 1 x 5 Blur, 40 dB BSNR

a. degraded image   b. minimum state

c. 1 x 6 x 6 model   d. 10 pixels added to left

a	b
c	d

In the case of the 3 x 3 blur, results were more significant. Although the mean-square improvements over the minimum case were slightly less than those for the 1 x 5 blur, the visual results were much more apparent. The minimum state case had noticeably more ringing artifacts in the lower spatial frequency areas of the background than did the cases with the model and/or states extended to the left. The two noise cases exhibited similar results. Examples of restored images are shown in figure 2.7.

Therefore, it would seem that the choice of model and state size to use depends on the type of blur. In the simpler one-dimensional 1 x 5 blur, the improvement obtained by adding model parameters and/or states does not seem to merit the extra computations required. However, for a more severe blur such as the 3 x 3 blur, the improvement may justify use of the larger state. The choice depends on the particular situation requiring filtering. For example, if speed is very important, it may be acceptable to use the lower order state, sacrificing some image quality to satisfy time constraints.

2.4 SUN Implementation

Both the Image Processing Lab (IPL) and the Robotics and Automation Lab (RAL) at Rensselaer Polytechnic Institute have purchased SUN workstations for use in research. It was therefore an additional goal of this project to implement a user-friendly, window-based image processing tool for use on the SUNs. The results of the above tests were used in writing the software for the ROMKF part of the program.

The basis of the program is a public domain image processing program called Imagetool, written at the National Center for Supercomputing Applications at the University of Illinois by Norman and Song [9]. The original program, written in Sunview, has utilities for displaying images, cutting and pasting parts of images, zooming, colormap adjustment, graphing, and file transferral. The major modification was to replace the file transfer option with a filter option containing programs

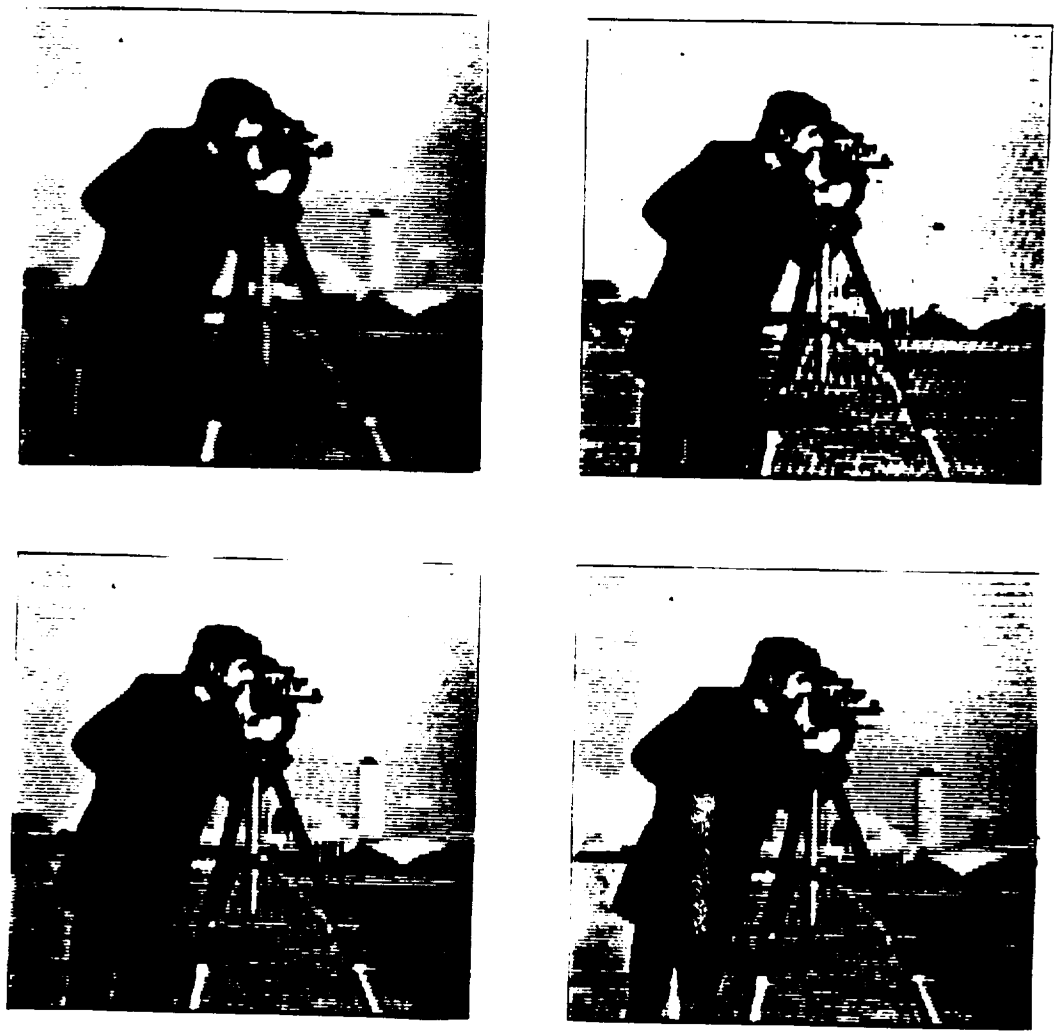


Figure 2.7: Comparison of Minimum State with Cases with Maximum Improvement - 3 x 3 Blur, 40 dB BSNR

a. degraded image	b. minimum state	a	b
c. 1 x 6 x 6 model	d. 5 pixels added to left (3 rows)	c	d

for image restoration. Input is obtained from the user through various pop-up windows, activated by menus or buttons.

Included in these utilities are the following:

- Least -squares parameter identification, with the option of bias compensation, as suggested by Woods and Ingle [13].
- Degradation routines for adding known blur and noise.
- The ROMKF
- Motion estimation.

The routines for the ROMKF were adapted from those written by Angwin [2]. Due to the results of the state/model tests, the filter is set up for an image model of size 1 x 1 x 1, with the option of adding additional states to the left if desired. The motion estimation programs were adapted from programs written by Naveen and Kim [5]. The specific details of the motion estimation algorithm are given in Part 3.

PART 3
Motion Estimation

3.1 Introduction

The optic flow algorithm used was one adapted from Netravali and Robbins [8] by Woods and Naveen [12]. This method uses the pel recursive algorithm of Netravali and Robbins [8] in a hierarchical (pyramid) structure. The use of pyramids in motion estimation allows for the computation of large velocities which might not be detectable by nonheirarchical methods. In a pyramid-based algorithm, the image is subsampled, using lower spatial frequencies to obtain the first velocity estimate which is then refined by the use of higher and higher frequencies. The specifics of the method used here will now be given.

3.2 Theory

The motion of an object can be represented by the following equation:

$$I_N(\boldsymbol{x}) = I_{N-1}(\boldsymbol{x} - \delta(\boldsymbol{x})) \quad \text{for all } \boldsymbol{x}, \tag{3.1}$$

where I_{N-1} and I_N are frames N-1 and N of the image sequence, $\boldsymbol{x}$ is the vector indicating the position of the pixel in a frame, and $\delta(\boldsymbol{x})$ is its displacement. The goal of the motion estimator is to determine the displacement $\delta(\boldsymbol{x})$. Here, the expression for the displacement is derived in [8] through linear regression as the following iterative equation:

$$\delta(\boldsymbol{x})^{i+1} = \delta(\boldsymbol{x})^i - \varepsilon \cdot \mathrm{DFD}(\boldsymbol{x}, \delta(\boldsymbol{x})^i) \cdot \nabla \mathrm{I}_{N-1}(\boldsymbol{x} - \delta(\boldsymbol{x})^i), \tag{3.2}$$

where DFD is the displaced frame difference, and is given by

$$\mathrm{DFD}(\boldsymbol{x}, \delta(\boldsymbol{x})) = I_N(\boldsymbol{x}) - I_{N-1}(\boldsymbol{x} - \delta(\boldsymbol{x}))\,, \tag{3.3}$$

the ∇ is the gradient with respect to $\boldsymbol{x}$, and i is the iteration count. The ε can be taken to be a constant, but for better performance, the following was used as suggested by Walker and Rao [10] :

$$\varepsilon = \frac{1}{2 \cdot |\nabla I_{N-1}(\boldsymbol{x} - \delta(\boldsymbol{x})^i)|^2} \tag{3.4}$$

For this implementation, three iterations are run for each pixel. Bilinear interpolation is used to compute the DFD for non-integral values of $\delta(\boldsymbol{x})$, and two adjacent pixels are used to evaluate ∇I as follows:

$$\text{x-component of } \nabla I = (\mathrm{I}_b - \mathrm{I}_a)/2 \tag{3.5}$$

$$\text{y-component of } \nabla I = (\mathrm{I}_u - \mathrm{I}_l)/2, \tag{3.6}$$

where I_b and I_a are the two pixels before and after the current pixel, and I_u and I_l are the upper and lower adjacent pixels.

As previously mentioned, the algorithm above can have poor performance at higher velocities. Therefore, it is used in a quadrature mirror filter (QMF) pyramid. The elements of a K level pyramid of an image I_N can be represented as $I_N^0, I_N^1, ..., I_N^{K-1}$ and I_N^K, where I_N^0 is the original image, I_N^1 is subband 11 (low-low) of I_N^0, I_N^2, is subband 11-11 of I_N^0, etc. So, the image at a given level of a pyramid is subband 11 of the image just below it. Therefore, I_N^K is the image with the least spatial frequencies. This is illustrated in figures 3.1 and 3.2.

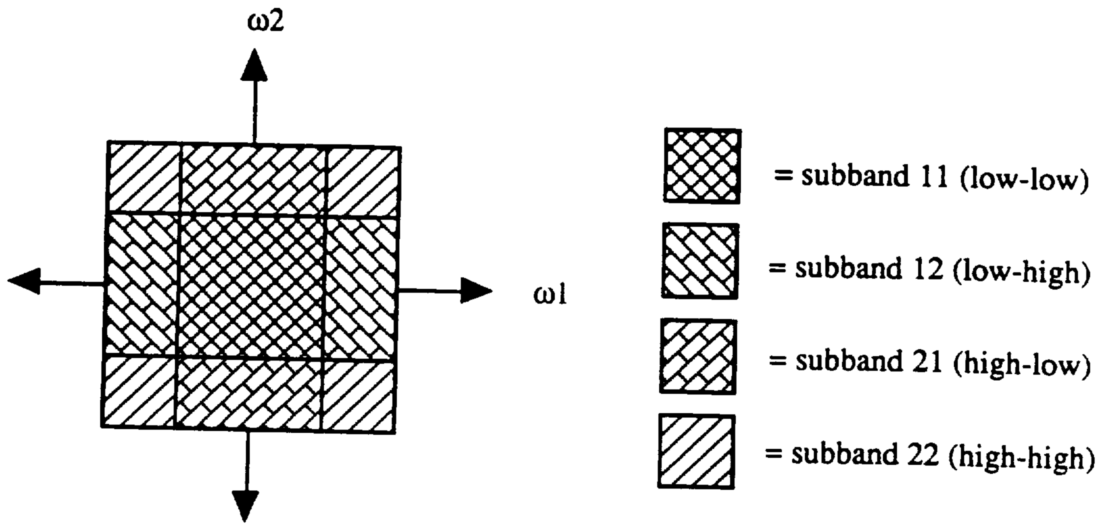


Figure 3.1: Subbands Used in the QMF Pyramid

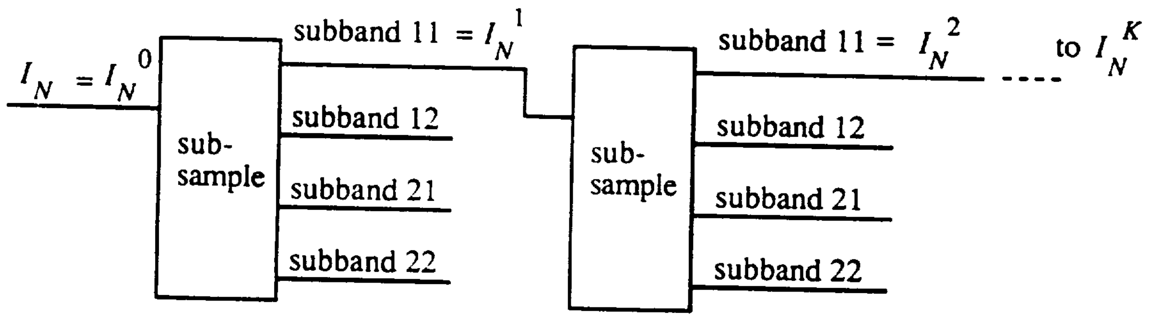


Figure 3.2: Pyramid Construction

Starting with the K-th level, the displacement $\delta(\boldsymbol{x})$ between I_N and I_{N-1} is obtained using the pel recursive algorithm. Then this estimate is expanded to obtain a coarse estimate which corresponds to the next level images which are four times larger. This coarse estimate is used as the initial estimate of motion between I_N^{K-1} and I_{N-1}^{K-1} in order to get the displacement at the K-1 level. This process of refining the initial estimate is repeated until we get the displacement between the full resolution images I_N^0 and I_{N-1}^0.

In their algorithm, Woods and Naveen [12] also blur (lowpass filter) the displacements at each pyramid level in order to remove noise caused by the pixel recursive estimation. However, in this implementation, blurring of the displacements was not done in order to observe the effect of the ROMKF alone.

3.2 Experimental Results

As described in the introduction, the main goal of this thesis is to examine the results of applying the ROMKF to motion estimation of degraded sequences. The ROMKF was used in two ways: to filter the degraded sequence prior to the motion estimation, and to filter the displacement vectors resulting from unrestored sequences. Two types of image sequences were used in the experiments: a set of computer-generated sequences with known velocities, and a scanned image of a robot arm, moving with unknown velocity. Each sequence consisted of two images. The artificial images were obtained using the following equation from [8]:

$$I(\boldsymbol{x},t) = \begin{cases} 127 \text{ if } \|\boldsymbol{R}\| > 100, \\ 127(1 + e^{-.05\|\boldsymbol{R}\|} \cos(2\cdot\pi\cdot\|\boldsymbol{R}\|)), \text{ otherwise,} \end{cases} \tag{3.7}$$

where $\boldsymbol{R} = \boldsymbol{x} - (\,x_0 + \boldsymbol{D}t\,)$, 127 is the background intensity on a scale of 0 - 255, $\boldsymbol{D}$ is the displacement of the pattern per frame, and $\|\cdot\|$ denotes the Euclidian norm. The resulting image is a series of alternating light and dark concentric rings with exponentially decreasing radial intensity variation. The first image (with zero displacement) is shown in figure 3.3.

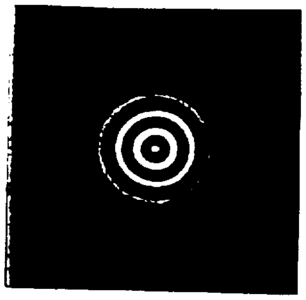


Figure 3.3: Artificial Circle Image

Six velocities were used in generating artificial sequences: two, four, and ten pixels per frame in the positive x direction only (to the right), and two, four, and ten pixels per frame in both the x and y directions (down and to the right).

The robot arm sequence was obtained from the RPI robotics lab. The first image of the sequence is shown in figure 3.4 . The exact velocity is unknown. However, it is uniform and estimated by visual inspection to be approximately four pixels per frame in the x direction and twelve pixels per frame in the y direction (down and to the right).

Prior to implementing the motion estimation algorithm, tests were done to determine the optimal number of pyramid levels for each velocity. This was determined by calculating the mean-square-error between the second image of the sequence and a prediction of the second image which was obtained using the displacements from the motion estimation algorithm..The results, shown in table 3.1, show the effectiveness of using the pyramid structure. From these results, it was decided to use one level for the two

Figure 3.4: Robot Arm Image

and four pixel velocities (both x and x & y directions), three levels for both of the ten pixel velocities, and four levels for the robot arm sequence.

For the undegraded sequences, it is possible to measure the accuracy of the displacements by the above approach of predicting the next frame. However, this is not the case when the sequence is degraded. In this situation, the predicted image is formed using the restored displacements which, being closer to the true displacement, should give an improved prediction. Therefore, comparing the second image of the degraded sequence with the predicted (hopefully undegraded) image will not be useful. Therefore, it was decided to compare the displacements resulting from the filtering with benchmark displacements calculated from the sequences before degradation.

A second task that was necessary prior to filtering the images and displacements was to determine the model coefficients $c_{k,l}$ and the plant noise variance σ_w^2 in equation 2.1. These parameters were calculated using least squares parameter estimation, assuming a 1 x 1 x 1 model. The undegraded images and benchmark displacements were

Table 3.1: Results of Varying the Number of Pyramid Levels

Image	Number of Levels	MSE
Circle - 2 pixels vel	0	3.14
	1	1.40
	2	2.98
Circle - 4 pixels vel	0	16.57
	1	4.24
	2	5.01
Circle - 10 pixels vel	1	19.68
	2	19.17
	3	16.10
	4	27.66
Robot Arm	0	441.29
	1	260.06
	2	123.84
	3	73.66
	4	52.97
	5	85.95

used for the computations. The results are shown in table 3.2, with the coefficient positions shown in figure 3.5. The coefficients were identical for all frames of the circle image, since the circles only shifted on the background, with all of the circle still in the frame. Note that the models for the images are not the same as the models for the displacements. It is assumed because of linearity, however, that the same degradation model can be used for the displacement and the images in the sequence. Therefore, the filtering of the displacements takes the same form as for images, the difference being in the model parameters.

Three cases for the degradation were used: 20 dB noise, 1 x 5 known linear blur, and the combination of the 1 x 5 blur with 20 dB BSNR. The blur support is the same as that in figure 2.5. Comparisons are calculated in decibels in a manner similar to the MSE η_{dB} used in part 2, and are tabulated in tables 3.3 - 3.5. The degraded MSE (between the benchmark and the displacement from the degraded sequence) is also provided to give an indication of the severity of the error induced by the degradation.

For the case with noise alone, improvement was obtained for most cases by filtering the sequence, and in all cases by filtering the displacements. Better performance was obtained in the latter case, especially at lower velocities. The degraded MSE results show the increasing effects of the noise as the velocity increases, corresponding to a decrease in the improvement obtainable by the filtering.

In the case with blur alone, neither method produced improvement, but rather increased the displacement error. Filtering of the displacements produced the most error. It is also seen from the degraded MSE that, especially at the lower velocities of two and four pixels, the effects of the blur on the displacements were less than those from noise alone.

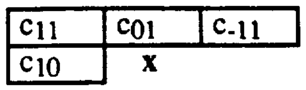


Figure 3.5: NSHP Support

Table 3.2: Filter Parameters

Image/Disp.	c_{11}	c_{01}	c_{-11}	c_{10}	σ_w^2
Circle Image	-.8119	.8128	.0696	.9364	4.7420
2 pel x disp. - hor.	-.1062	.6227	.0123	.3198	.1374
- vert.	-.0190	.3584	.1975	.4039	.0861
2 pel xy disp. - hor.	-.2899	.5470	.1735	.4403	.1989
-vert.	-.2895	.2770	.2130	.6513	.1980
4 pel x disp. - hor.	-.1029	.4121	.1529	.4627	.2694
- vert.	-.2296	.7096	.0412	.3610	.4242
4 pel xy disp. - hor.	-.2927	.6139	.1060	.3986	.4769
- vert.	-.2899	.2488	.1719	.6797	.4695
10 pel x disp. - hor.	-.3704	.4223	.2149	.7388	.8096
- vert.	-.4037	.4461	.2148	.7487	.8615
10 pel xy disp.-hor.	-.3834	.4484	.2119	.7284	.9909
-vert.	-.3812	.4328	.2164	.7370	.9897
Robot arm - img 1	-.5345	.5922	.1368	.8060	11.1393
Robot arm - img 2	-.5827	.6497	.1084	.8251	12.2755
Robot disp. - hor.	-.3372	.3787	.2430	.7208	1.5248
Robot disp. - vert.	-.3480	.3889	.2450	.7173	1.7415

Table 3.3: Results With 20 dB Noise

	Degraded	MSE Improvement η_{dB}	
Sequence	MSE	Rest. Seq.	Rest. Disp.
2 pel x - horizontal	1.02	.17	4.77
- vertical	1.04	.49	7.87
2 pel xy - horizontal	1.05	.13	4.19
- vertical	1.08	.38	4.31
4 pel x - horizontal	1.12	.16	2.10
- vertical	1.10	.33	4.18
4 pel xy - horizontal	1.21	.11	2.63
- vertical	1.21	.30	2.90
10 pel x - horizontal	22.06	-.35	.19
- vertical	21.59	.45	.45
10 pel xy - horiz.	24.94	-.34	.38
- vertical	28.88	.37	.35
Robot arm - horiz.	38.56	.41	.59
- vertical	36.36	.06	.20

Table 3.4: Results With 1 x 5 Blur

Sequence	Degraded MSE	MSE Improvement η_{dB} Rest. Seq.	Rest. Disp.
2 pel x - horizontal	.12	-.97	-9.54
- vertical	.09	-2.76	-14.35
2 pel xy - horizontal	.13	-.90	-13.59
- vertical	.13	-1.41	-11.83
4 pel x - horizontal	.22	-.19	-15.86
- vertical	.17	-.71	-12.35
4 pel xy - horizontal	.22	-1.63	-15.57
- vertical	.21	-1.55	-12.79
10 pel x - horizontal	17.50	-.34	-2.08
- vertical	17.54	-.60	-1.92
10 pel xy horiz.	22.41	-.52	-1.81
- vertical	24.03	-.32	-1.63
Robot arm - horiz.	28.41	-.15	-1.81
- vertical	25.94	-.44	-2.11

Table 3.5: Results With 1 x 5 Blur Plus 20 dB Noise

	Degraded	MSE Improvement η_{dB}	
Sequence	MSE	Rest. Seq.	Rest. Disp.
2 pel x - horizontal	1.04	-.04	4.37
- vertical	.97	-.22	7.56
2 pel xy - horizontal	1.08	.08	3.90
- vertical	1.01	-.21	3.92
4 pel x - horizontal	1.21	0.00	1.53
- vertical	1.08	-.31	3.71
4 pel xy - horizontal	1.25	.11	2.05
- vertical	1.15	-.11	2.61
10 pel x - horizontal	24.26	.13	-.04
- vertical	20.66	.33	.24
10 pel xy - horiz.	29.50	.56	.11
- vertical	29.22	.09	.09
Robot arm - horiz.	38.54	.17	.57
- vertical	37.04	.12	.21

Finally, for the case with blur and noise, results were more encouraging. Improvement was obtained in all cases where the displacements were filtered, and in about half of the cases with the sequences filtered. As in the other two degradation cases, the results for filtering the displacements were better at lower velocities. However, the reverse was true for filtering the sequences: the worst cases were those with lower velocities. The degraded MSE results were similar to those with noise alone.

In addition to using the MSE improvement as a measurement of effectiveness, it was also useful to have a visual means of examining the displacements. This was done by scaling the displacements and displaying them as SUN raster images. The scaling was necessary since the image format truncates the data to integer format. The degraded and filtered images are also provided for inspection.

Figure 3.6 shows the first images in the degraded and restored circle sequence. There is little apparent difference between either the degraded and original images, or the degraded and restored images, for all the degradation cases. However, we know that the degradations do have an effect on the displacement calculations, especially at high velocities. So, even when degradations are not visually obvious, the displacements can still be affected, particularly at high velocities.

The displacements for the circle sequence are shown in figures 3.7-3.9. For the noise alone, the effect is clearly seen in the degraded sequence displacements. The restored sequence displacements and restored displacements are not visibly much improved. The same is true for the blur plus noise case. It appears that the noise effects are the most prominent, which is as expected, since the degraded MSE results showed that noise had a greater effect than blur for the four pixel velocity.

The displacements from blur alone are more interesting. As expected from the decibel improvements, the degraded displacements are not much different from the

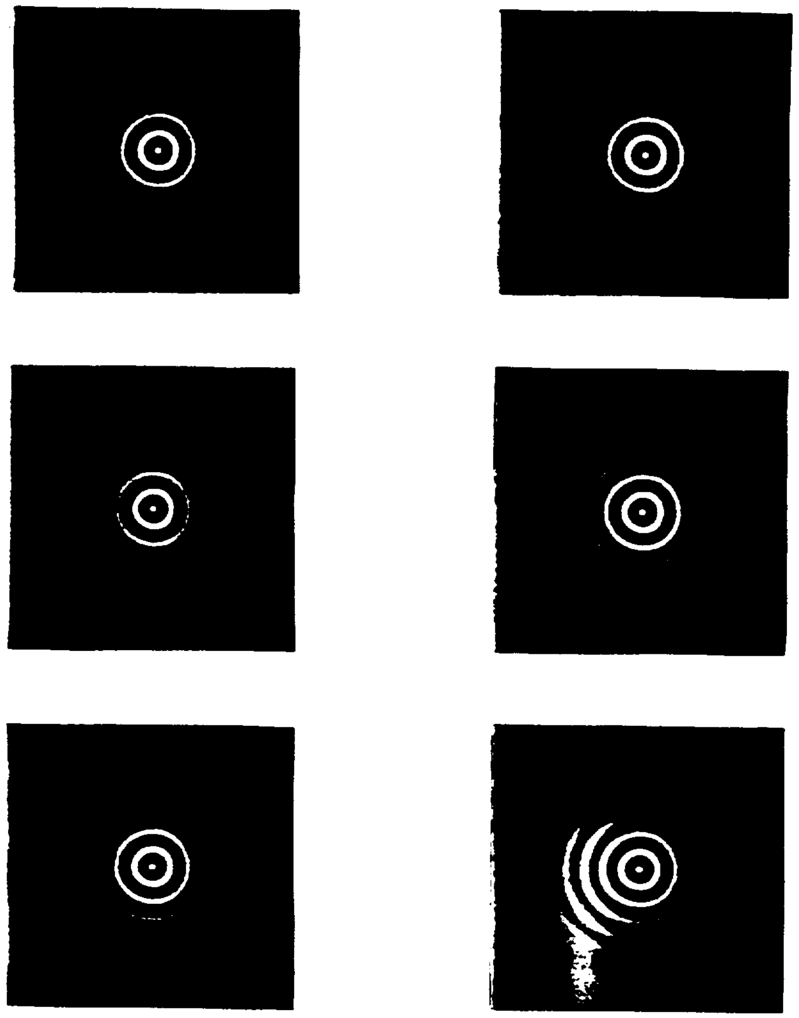


Figure 3.6: Degraded and Restored Circle Images

a. 20 dB Noise - Degraded	b. 20 dB Noise - Restored
c. 1 x 5 Blur - Degraded	d. 1 x 5 Blur - Restored
e. Blur & Noise - Degraded	f. Blur & Noise - Restored

a	b
c	d
e	f

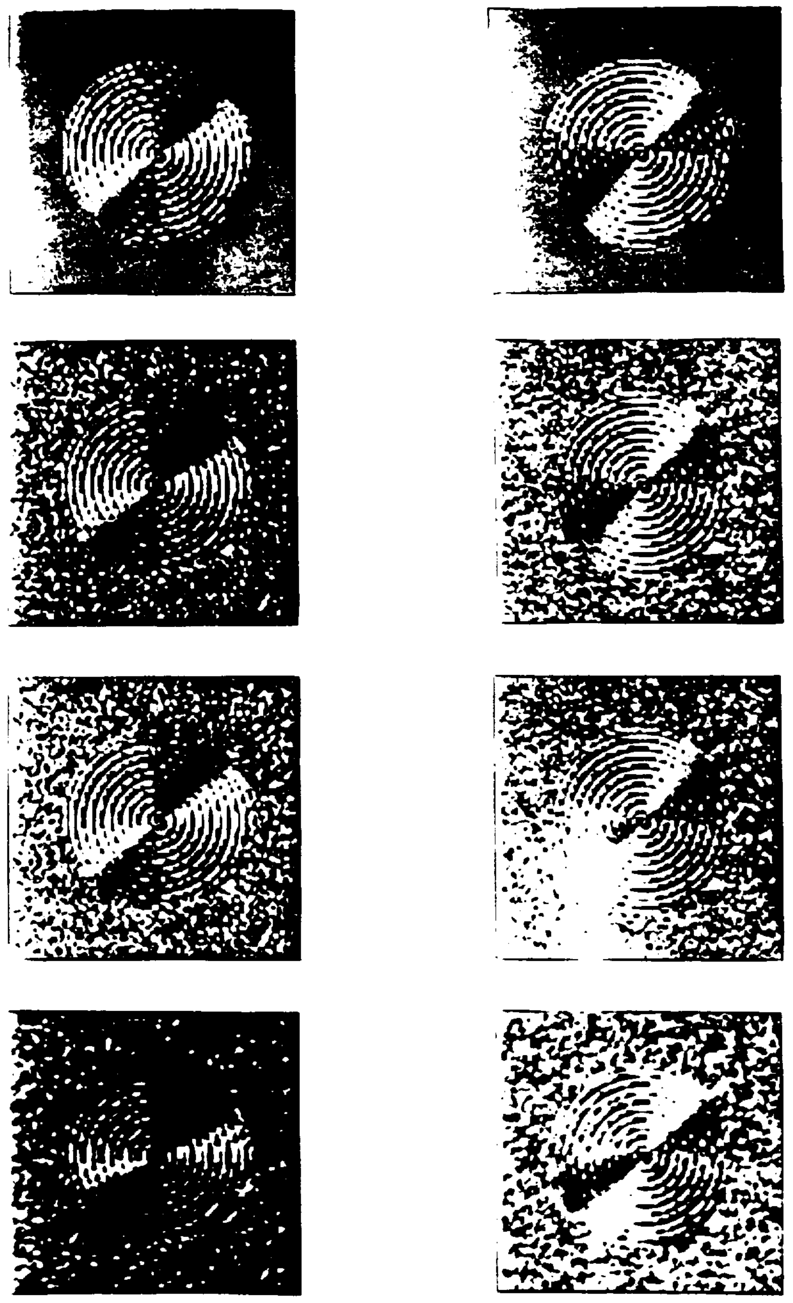


Figure 3.7: Circle Displacements for 20 dB Noise

a. Benchmark Displacement - horizontal b. Same - vertical
c. Degraded Sequence Disp. - horizontal d. Same - vertical
e. Restored Sequence Disp. - horizontal f. Same - vertical
g. Restored Displacement - horizontal h. Same - vertical

a	b
c	d
e	f
g	h

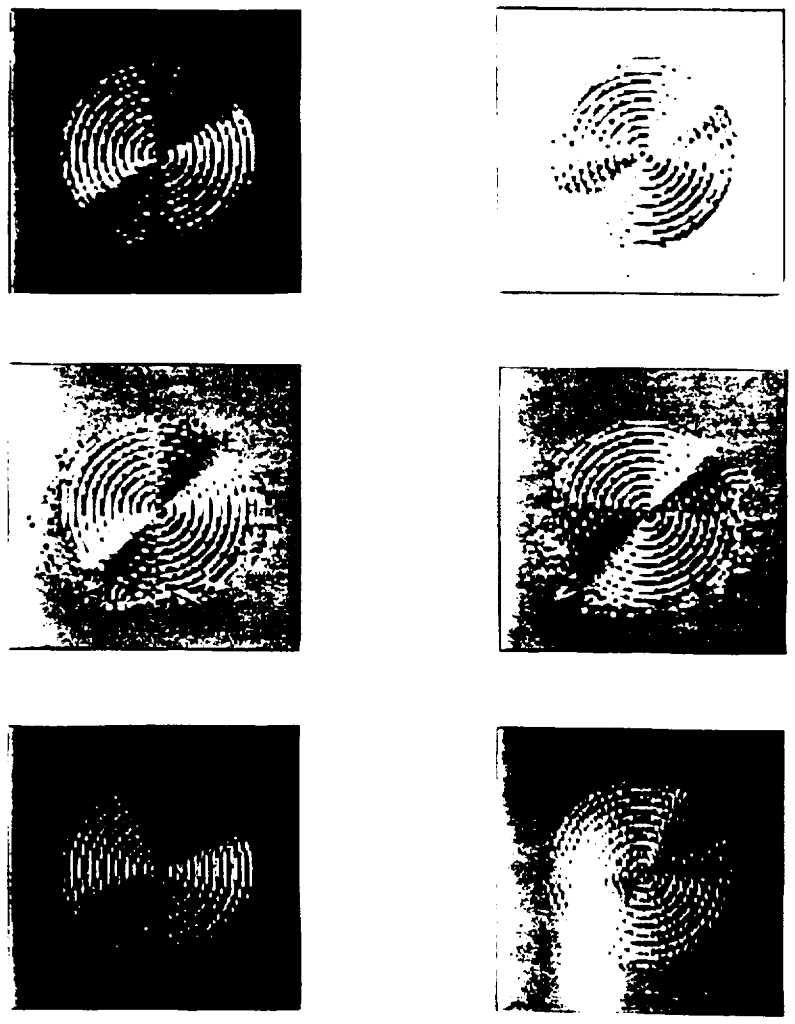


Figure 3.8: Circle Displacements for 1 x 5 Blur

a. Degraded Sequence Disp. - horizontal b. Same - vertical

c. Restored Sequence Disp. - horizontal d. Same - vertical

e. Restored Displacement - horizontal f. Same - vertical

a	b
c	d
e	f

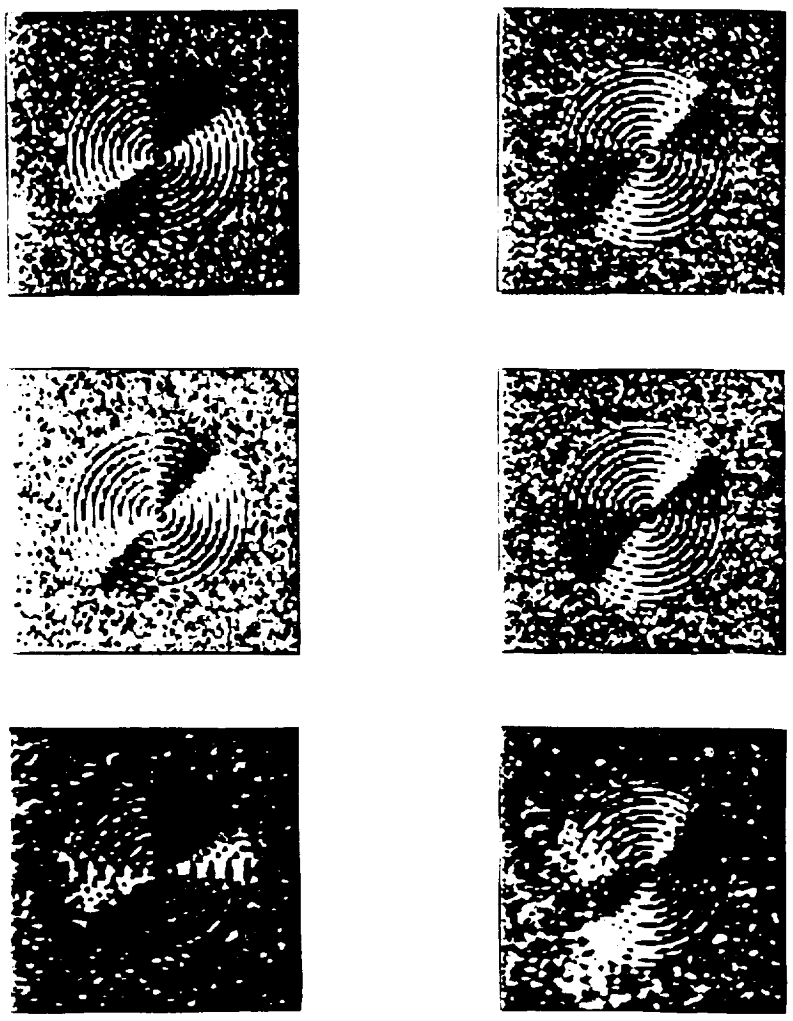


Figure 3.9: Circle Displacements for 1 x 5 Blur plus 20 dB Noise

a. Degraded Sequence Disp. - horizontal b. Same - vertical

c. Restored Sequence Disp. - horizontal d. Same - vertical

e. Restored Displacement - horizontal f. Same - vertical

a	b
c	d
e	f

benchmarks. The restored displacements from both methods show more visible changes, however. In the case of the restored sequence displacements, errors are seen to be concentrated in the area of the circle edges. This seems to indicate that one cause for the lack of improvement over the degraded sequence displacements could be filtering artifacts which are often present near edges. In the case of the restored displacements, the lack of improvement is also clearly shown. The filter appears to be overcompensating in some fashion. One possible explanation is that the blur model for the displacements does not correspond to that of the images in the sequence, as previously assumed.

After these first trials were completed, further testing was done to see if improvements could be obtained in those cases where none was before. The visual results for the 1 x 5 blur were helpful in trying to improve that case. Since the error appeared to be concentrated just outside the boundary of the circle, where the velocity should be zero, it was decided to try thresholding the benchmark displacements, and only use displacements above the threshold in the comparisons. In this way, the erroneous data outside the circle is not considered. The results of thresholding did support this assumption for most of the cases. Numerical results are presented in table 3.6. Note that the degraded MSE results shown here are larger than those obtained when the error is calculated over the entire field (table 3.4). This shows the localization of the error near the moving circle.

Thresholding was also tried with some of the displacements that were filtered directly. While the error did decrease with the thresholding, it did not improve beyond that of the degraded MSE.

A method discovered for improving the results for the filtered displacements was to consider the error as noise, and so include a noise variance in the filter. This was only successful for the ten pixel velocities. Improvements were obtained for these sequences in the case of blur alone and blur plus noise, using a noise variance of 20 for the blur alone

Table 3.6: Results of Thresholding the Circle Displacements (Rest. Sequence), 1 x 5 Blur Case

Image Sequence	Threshold	Degraded MSE	η_{dB}	Number of pts.
2 pel x - horizontal disp.	.2	.41	1.98	7207
- vertical disp.	.2	.29	1.20	7392
2 pel xy - horizontal disp.	1.0	.58	1.40	3604
- vertical disp.	1.0	.60	2.22	3604
4 pel x - horizontal disp.	.5	.84	1.32	6088
- vertical disp.	.5	.59	1.58	6268
4 pel xy - horizontal disp.	1.0	.97	.14	9065
- vertical disp.	1.0	.98	1.40	9235
10 pel x - horizontal disp.	9.9	89.04	-.08	1844
- vertical disp.	9.9	87.70	.45	1457
10 pel xy - horizontal disp.	9.9	156.09	-.08	4286
- vertical disp.	9.9	142.66	.45	4286

and 21.5 for blur plus noise, as opposed to using a variance of zero. DB results are seen in table 3.7. These results can be compared with those in tables 3.4 and 3.5.

In the cases where more than one pyramid level was used, as in the ten pixel velocities and the robot arm sequence, the visual results were not as clear. Due to the increased number of levels, and also the scaling, the displacement images are much more blurred in appearance, and the edges of motion cannot be seen. Figure 3.10 shows the results for the improved ten pixel xy velocity sequence using the extra noise variance, along

Table 3.7: Filtered Displacement Improvements

Sequence	MSE Improvement η_{dB} Blur Alone	Blur + Noise
10 pel x - horizontal disp.	.39	1.05
- vertical disp.	.88	1.32
10 pel xy - horizontal disp.	.88	1.30
- vertical disp.	.87	1.12

with the benchmark, degraded and unimproved cases. While the images themselves are not very meaningful, it is possible to see the differences caused by the filtering. In the case where no noise is assumed in the filter. the displacements have a much less smooth appearance when compared with the benchmark images. In the case where the noise variance was set at 20, the images have a smoothed appearance closer to that of the benchmarks.

Figure 3.11 shows examples of the degraded and restored robot arm images. The visual differences between the original, degraded , and restored images are slightly more apparent than in the circle images, but are still not great. As in the ten pixel velocity circle sequence, the robot arm displacements, as displayed visually, are quite blurred and unclear. Displacements from the 20 dB noise case are shown in figure 3.12. Improvements were attempted as with the circle sequences, but without success.

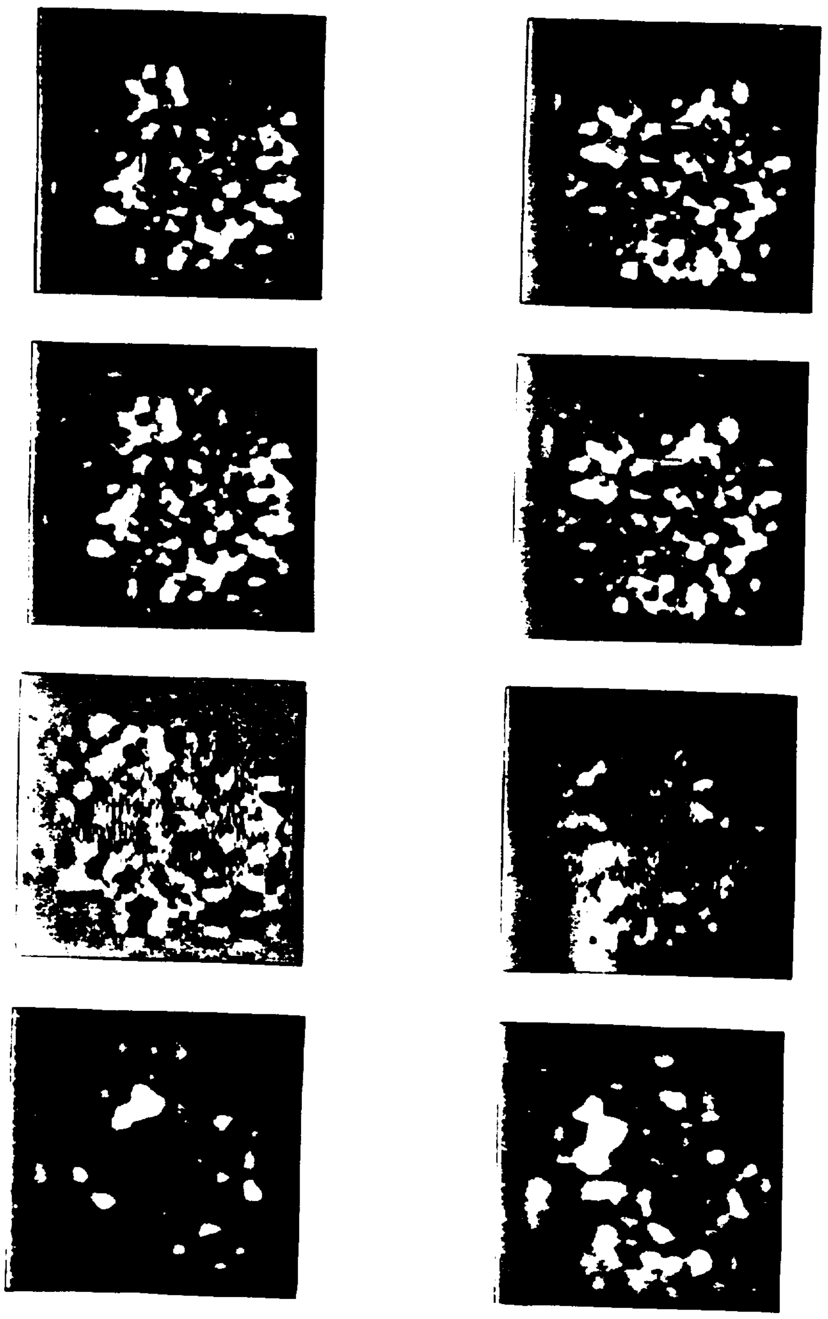


Figure 3.10: Displacements for 10 Pixel xy Velocity, 1 x 5 Blur

a. Benchmark Displacement - horizontal b. Same - vertical
c. Degraded Sequence Disp. - horizontal d. Same - vertical
e. Restored Disp. no noise in filter - horiz. f. Same - vertical
g. Restored Disp. - noise var.= 20 - horiz. h. Same - vertical

a	b
c	d
e	f
g	h

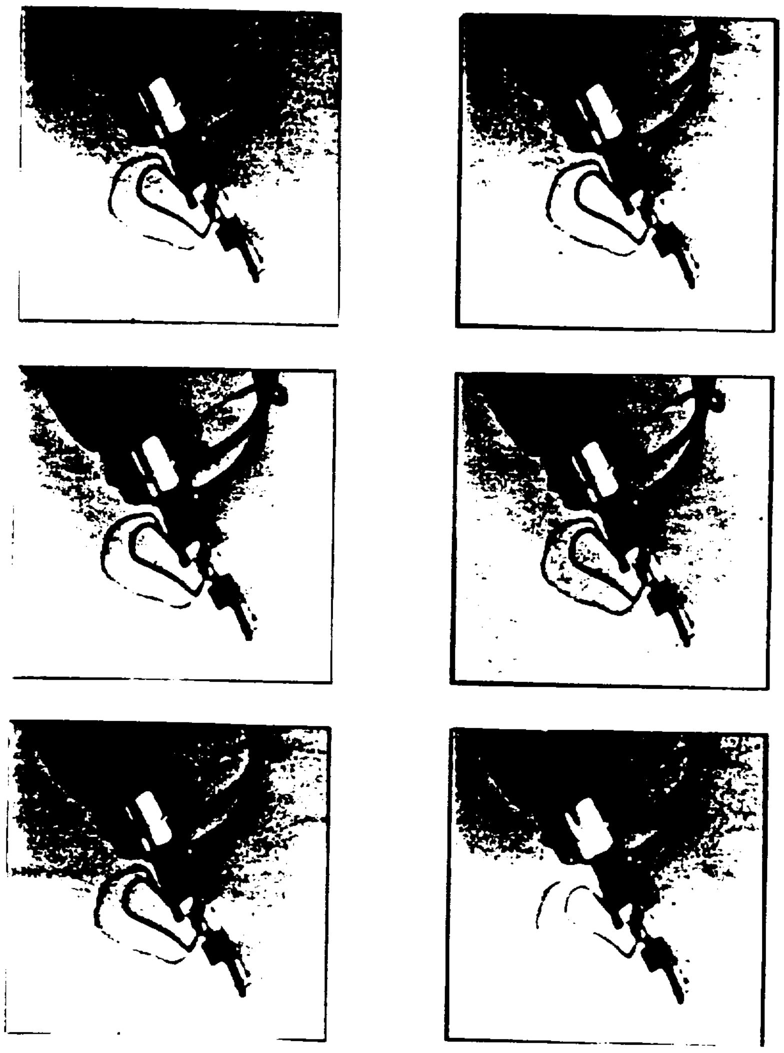


Figure 3.11: Degraded and Restored Robot Arm Images

a. 20 dB Noise - Degraded   b. 20 dB Noise - Restored

c. 1 x 5 Blur - Degraded   d. 1 x 5 Blur - Restored

e. Blur & Noise - Degraded   f. Blur & Noise - Restored

a	b
c	d
e	f

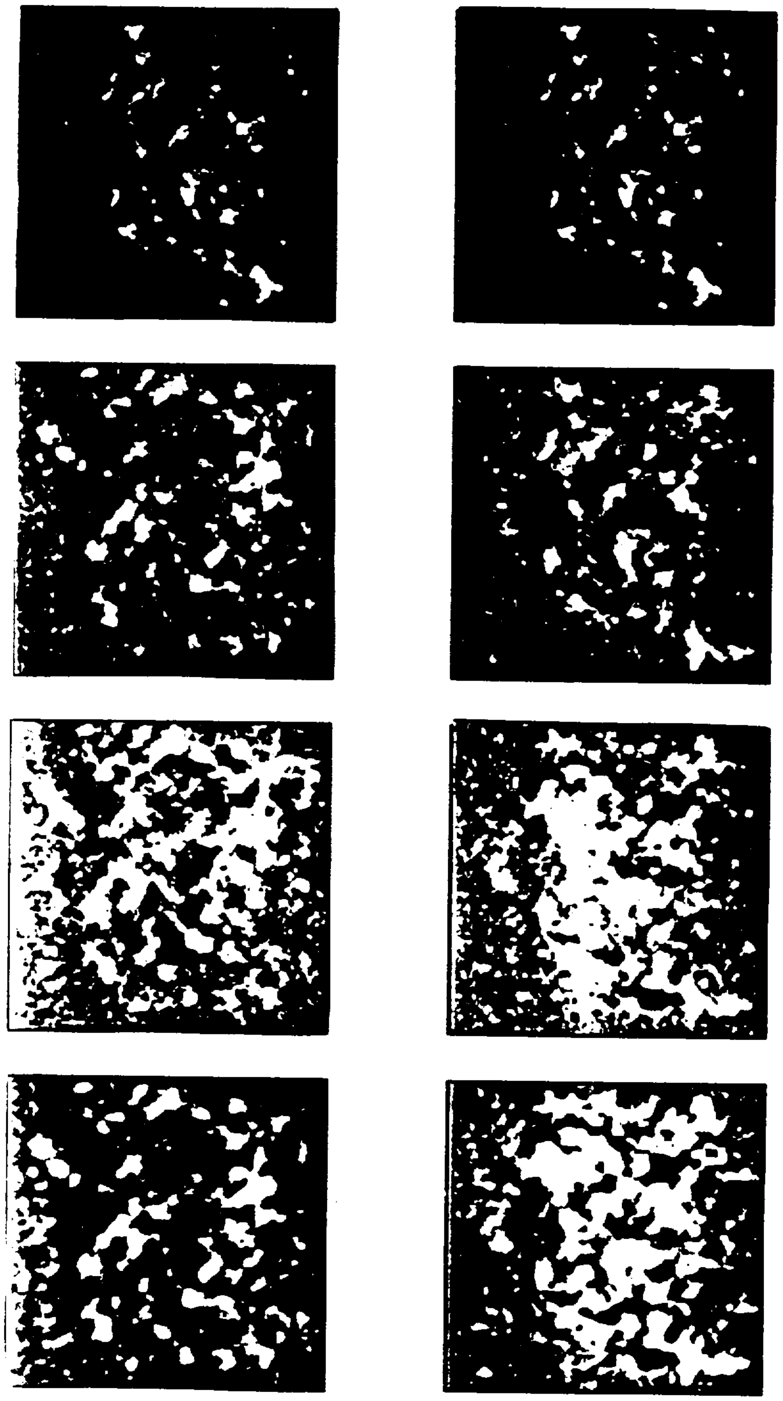


Figure 3.12: Robot Arm Displacements for 20 dB Noise

a. Benchmark disp - horiz. b. Same - vertical
c. Degraded seq. disp. - horiz. d. Same - vertical
e. Restored seq. disp. - horiz. f. Same - vertical
g. Restored disp.- horiz. g. Same - vertical

a	b
c	d
e	f
g	h

PART 4

Discussion and Conclusions

The goal of this thesis was to investigate the use of the reduced order model Kalman filter in reducing motion estimation errors due to image degradations, and to examine the implementation of the filter and motion estimation algorithms on the SUN workstation. In order to determine the form of the filter for the SUN implementation, preliminary testing was done to find the optimum state and model configuration. These tests showed that in general, increasing the state to the left either by increasing the model or by adding extra states with the minimum model leads to some improvement over the minimum state. This improvement varies depending on the kind of degradation.

In the motion estimation tests, two methods of applying the ROMKF were used: filtering the sequences prior to motion estimation, and filtering the displacements from the degraded sequences directly. The same degradation model was used initially for both cases. In general, filtering the displacements directly led to greater improvements. The results varied depending on the type of degradation. For additive noise alone, both methods gave some improvement for almost all sequences used. The improvements decreased as the velocities increased. With blur and noise, the restoration of the displacements gave improvements in almost all cases, while the other method was only effective in the higher velocity cases. In the cases of restored displacements, the improvements again decreased as the velocities increased. The reverse was true for the displacements from the restored sequences.

Finally, the results for the blur alone were quite interesting. Numerically, both methods increased the displacement error, with the restored displacements being the worst. Results here improved as the velocities increased for both methods. Visual representations of the displacements showed a possible cause for the poor performance. For the displacements from restored sequences, the error was seen to be concentrated at the object boundaries, which could be due to artifacts in the images from the filtering process. Thresholding the displacements when calculating the results showed improvements in many cases. For the restored displacements, it was possible to obtain improvements by modelling the displacement error as observation noise in the filter. Improvements were obtained for the cases with higher known velocities of ten pixels per frame. Applying this technique also yielded improvements for the case of blur plus noise.

Therefore, it appears that restoring the displacements using the ROMKF on the displacement fields directly can lead to some improvement when the sequence is degraded.by noise or blur plus noise. Restoring the images prior to motion estimation also leads to improvements, but to a lesser degree. Both methods fail for blur alone, but since most real images are noisy to some extent, this is an unlikely case anyway. Improvements were also obtained in some cases when the degradation model for the filtered displacements was varied.

The improvements were slight in most cases, so further work needs to be done in this area. For example, tests can be done with additional lowpass filtering of the intermediate displacements in the pyramid, as done by Woods and Naveen [12]. Another option is to further investigate varying the degradation model for the displacements. The state size for the filter can also be increased, as only the minimum support was used in these tests.

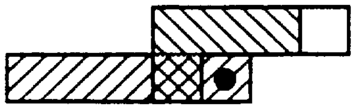
Another possibility is to use adaptive filtering. In these experiments, both the images and displacements were assumed to be spatially invariant. However, this is not a good assumption for real images and displacements. Therefore, the use of the adaptive ROMKF [2] ,which uses multiple models in moving windows, could have increased effectiveness. The ROMKF can also be extended to color sequences.in the same way that it was extended to color images [2].

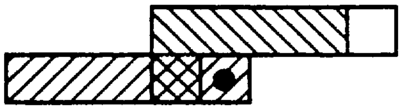
In addition to these proposals, other possibilities involve representing the sequence using augmented model and observation equations (equations 2.1 and 2.2) which would contain an added dimension representing the frame index, or time. By incorporating the displacements as well, simultaneous estimation of the displacements with the sequence can be done.

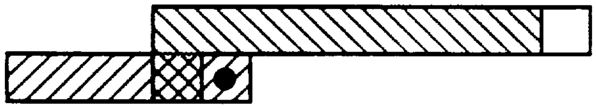
Finally, it was mentioned that in many applications, speed is important, as it is desirable to do the motion estimation in real-time. To increase the speed of the optic flow estimation and any error compensation used, the algorithms can be implemented on parallel machines such as the AMT Distributed Array of Processors (DAP).

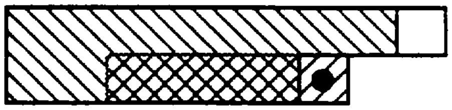
APPENDIX A

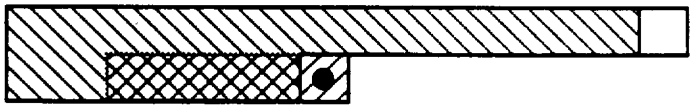
Model and Blur Supports

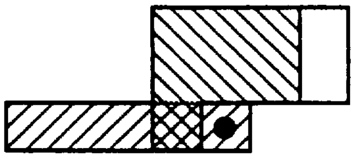

Case 1: 1 x 1 x 1 model
9 states

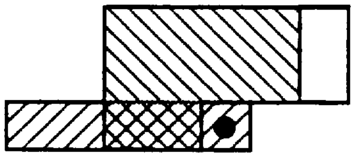

Case 2: 1 x 2 x 1 model,
10 states

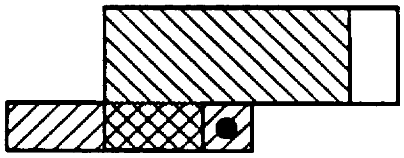

Case 3: 1 x 6 x 1 model,
14 states:

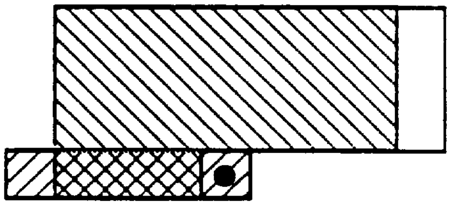

Case 4: 6 x 1 x 1 model
16 states:


Case 5: 6 x 6 x 1 model,
21 states:


Case 6: 1 x 1 x 2 model,
13 states:

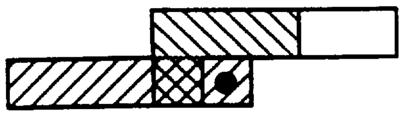

Case 7: 2 x 1 x 2 model,
15 states:

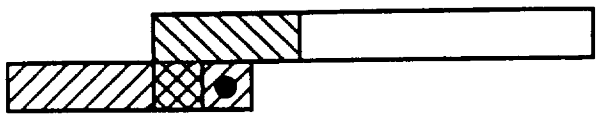

Case 8: 2 x 2 x 2 model,
17 states:

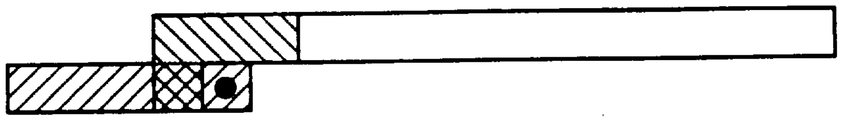

Case 9: 3 x 3 x 3 model, 29 states:

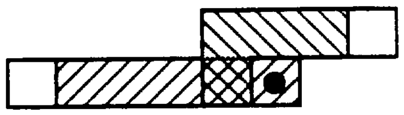
Key:
▧ = model support
▨ = blur support
● = $x(m,n)$

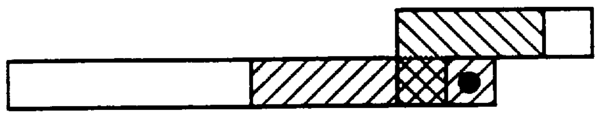
Figure A.1: 1 x 5 Blur Case, Model Varied

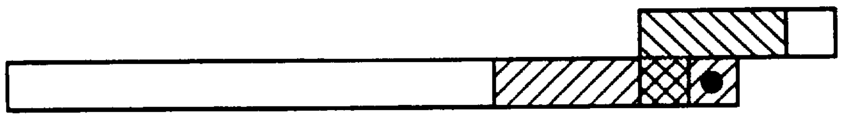

Case 1: 10 states

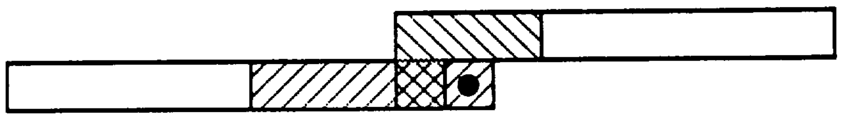

Case 2: 14 states


Case 3: 19 states


Case 4: 10 states


Case 5: 14 states


Case 6: 19 states


Case 7: 19 states

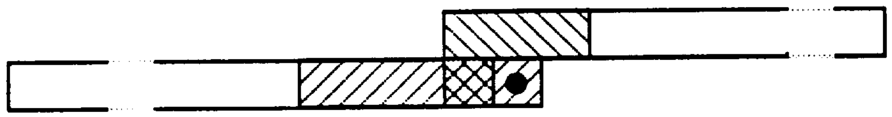
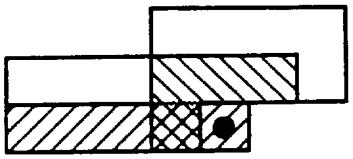
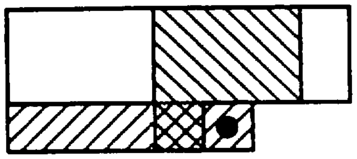
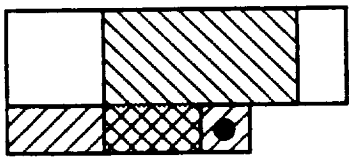

Case 8: 29 states

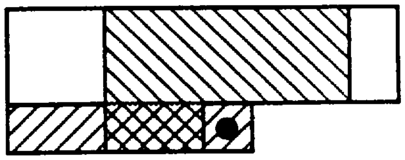
Figure A.2: 1 x 5 Blur Case, State Support Varied with 1 x 1 x 1 Model.

Note: For this and all following figures, the key is the same as for figure A.1.


Case 9: 16 states


Case 10: 1 x 1 x 2 model, 19 states


Case 11: 2 x 2 x 1 model, 19 states


Case 12: 2 x 2 x 2 model, 21 states

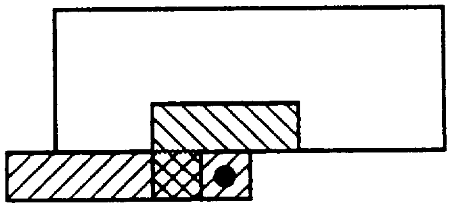
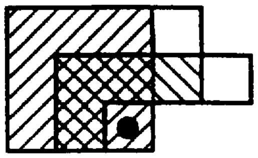
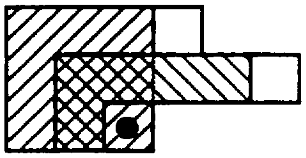

Case 13: 1 x 1 x 1 model, 29 states

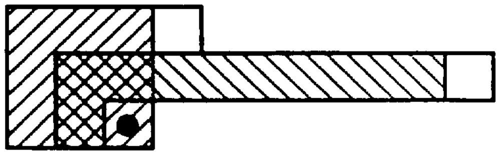
Figure A.3: 1 x 5 Blur Case, State Support Varied with 1 x 1 x 1 Model



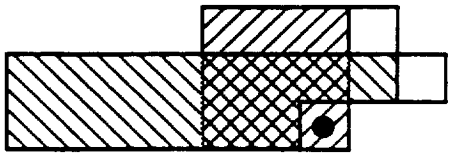
Case 1: 1 x 1 x 1 model
12 states



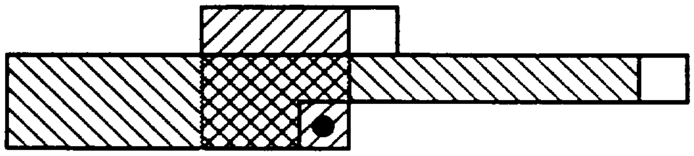
Case 2: 1 x 1 x 2 model
13 states



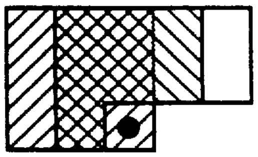
Case 3: 1 x 1 x 6 model
17 states



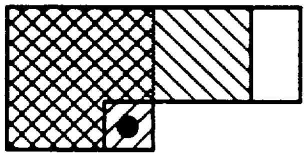
Case 4: 1 x 6 x 1 model
20 states



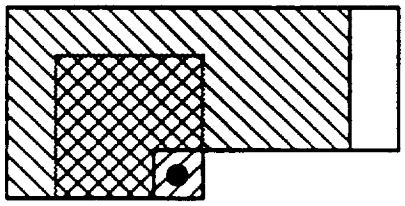
Case 5: 1 x 6 x 6 model
25 states



Case 6: 2 x 1 x 1 model
13 states

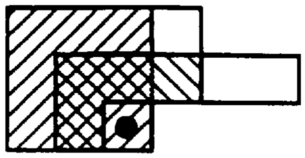


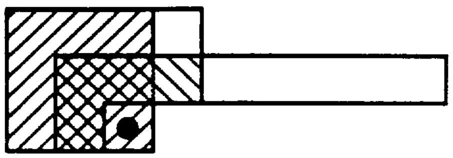
Case 7: 2 x 2 x 2 model
15 states

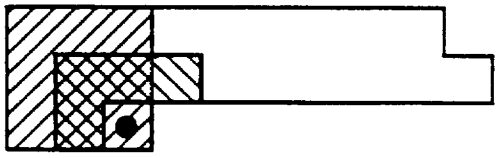


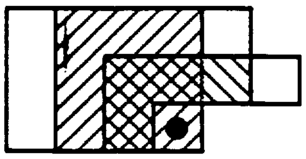
Case 8: 3 x 3 x 3 model
28 states

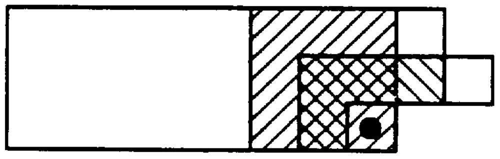
Figure A.4: 3 x 3 Blur Case, Model Varied

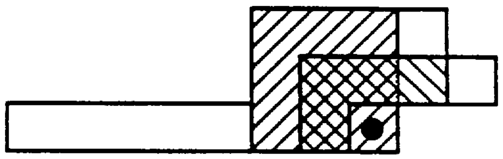

Case 1: 13 states

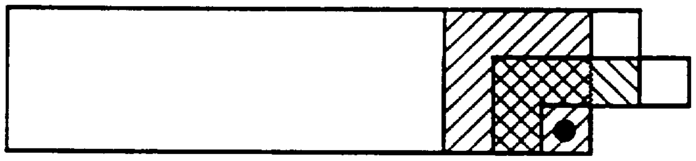

Case 2: 16 states


Case 3: 22 states


Case 4: 15 states


Case 5: 27 states


Case 6: 17 states


Case 7: 39 states

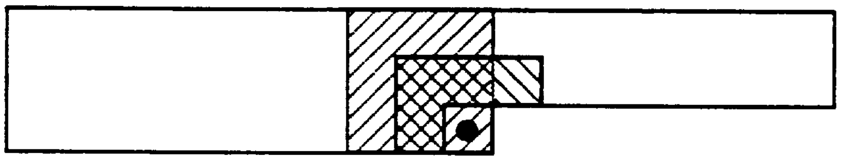

Case 8: 38 states

Figure A.5: 3 x 3 Blur Case, State Support Varied with 1 x 1 x 1 Model

APPENDIX B

The Kalman Filter Equations

The equations for the Kalman filter are given in this appendix. The derivation can be found in [4]. Here, the equations have been extended to two dimensions, with one direction of recursion.

The dynamic model for the state is represented as the following difference equation:

$$\underline{x}(m,n) = \boldsymbol{C}\underline{x}(m-1,n) + \boldsymbol{E}\underline{u}(m,n) + \boldsymbol{D}\underline{w}(m,n) \tag{B.1}$$

$$r(m,n) = \boldsymbol{H}\underline{x}(m,n) + v(m,n) \tag{B.2}$$

Given these models for x and r, the Kalman filter is defined as follows:

$$\underline{x}_b(m,n) = \boldsymbol{C}\underline{x}_a(m-1,n) + \boldsymbol{E}\underline{u}(m,n) \tag{B.3}$$

$$\underline{x}_a(m,n) = \underline{x}_b(m,n) + \boldsymbol{K}(m,n)[r(m,n) - \boldsymbol{H}\underline{x}_b(m,n)] \tag{B.4}$$

and

$$\boldsymbol{K}(m,n) = \boldsymbol{P}_b(m,n)\boldsymbol{H}^T\,[\boldsymbol{H}\boldsymbol{P}_b(m,n)\boldsymbol{H}^T + R]^{-1} \tag{B.5}$$

$$\boldsymbol{P}_b(m,n) = \boldsymbol{C}\boldsymbol{P}_a(m-1,n)\boldsymbol{C}^T + \boldsymbol{D}\boldsymbol{Q}\boldsymbol{D}^T \tag{B.6}$$

$$\boldsymbol{P}_a(m,n) = [\boldsymbol{I} - \boldsymbol{K}(m,n)\boldsymbol{H}]\boldsymbol{P}_b(m,n) \tag{B.7}$$

where

$$R = \mathrm{cov}(v) \tag{B.8}$$

and

$$Q = \mathrm{cov}(\underline{w})\,. \tag{B9}$$

In the above equations, $\boldsymbol{P}$ is the covariance matrix of the estimation error, and $\boldsymbol{K}$ is called the Kalman gain matrix. The equation for $\boldsymbol{K}$ is derived by minimizing the trace of $\boldsymbol{P}$. The above Kalman filter estimates the states by predicting the states and error covariance in (B.3) and (B.6), and using the gain $\boldsymbol{K}$ from (B.5) to update the states and error covariance in (B.4) and (B.7).

REFERENCES

[1] J.K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images - a review. *Proceedings of the IEEE*, 76(8): 917-935, August 1988.

[2] D.L. Angwin. *Adaptive Image Restoration Using Reduced Order Model Based Kalman Filters*. PhD thesis, Rensselaer Polytechnic Institute, Troy, New York, 1989.

[3] D.H. Ballard and O.A. Kimball. Rigid body motion from depth and optical flow. *Computer Vision, Graphics, and Image Processing*, 22: 95-115, 1983.

[4] A. Gelb, editor. *Applied Optimal Estimation*. The M.I.T. Press, Cambridge, MA, 1972.

[5] Y. Kim and T. Naveen. personal communication.

[6] A.K. Mahalanabis and K. Xue. An efficient two-dimensional chandrasekhar filter for restoration of images degraded by spatial blur and noise. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, 35(11): 1603-1610, November 1987.

[7] L. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. Report CMU-CS-87-185, December 1987.

[8] A.N. Netravali and J.D. Robbins. Motion-compensated television coding: part I. *The Bell System Technical Journal*, 53(3): 631-670, March 1979.

[9] M. Norman and C. Song. *Imagetool for the SUN Workstation, Version 1.0*. The University of Illinois, May 1988.

[10] D.R. Walker and K.R. Rao. Improved pel - recursive motion compensation. *IEEE Transactions on Communications*, 32(10): 1128-1134, October 1984.

[11] J.W. Woods and C.W. Radewan. Kalman filtering in two-dimensions. *IEEE Transactions on Information Theory*, 23: 473-482, July 1977.

[12] J.W. Woods and T. Naveen. Subband encoding of video sequences. Presented at *Advances in Intelligent Robotics Systems and Visual Communications and Image Processing*, Philadelphia, November 1989.

[13] J.W. Woods and V.K. Ingle. Kalman filtering in two-dimensions - further results. *IEEE Transactions on Information Theory*, 23: 473-482, July 1977.